Estructuras discretas
Práctica 05: Nuestras estructuras Pt. 1

Profesora: Alma Rosario Arévalo Loyola
Ayudante: José Ricardo Desales Santos
Ayudante: Karla Socorro García Alcántara
Laboratorio: Emiliano Galeana Araujo
Laboratorio: Rodrigo Guadalupe Chávez Jiménez

Facultad de ciencias, UNAM

Fecha de entrega: 4 de Diciembre de 2020

1 Introducción

Para esta práctica vamos a crear nuestra propia implementación de listas, ya hemos visto que las listas (En haskell) se pueden ver de la siguiente manera:

```
[]
(x:xs)
```

Esto, ya vimos que quiere decir que se puede ver como la lista que no tiene nada (La lista vacía) y la lista que tiene al menos un elemento y una lista. Todo bien, pero ¿Quién es la otra lista? (En el ejemplo `xs`) pues es la misma estructura, la cual puede ser la lista vacía en cuyo caso toda la lista habría acabado o puede cumplir con la segunda condición, que es que tenga un elemento y otra lista y así nos podemos seguir hasta que tengamos algo más intersante que hacer.

Como podemos ver, la definición que damos sobre listas es un poco extraña, ¿Cómo podemos definir a una lista como una misma lista? bueno no estamos definiendo a la lista como la misma lista, sino que le agregamos

un elemento y luego usamos la misma definición. Por ejemplo los números naturales podemos verlos como sigue:

1. El cero es natural.
2. Si n es natural, entonces *suc* n es natural.
3. Son todas.

Donde *suc* es la función sucesor, es decir dado un número, nos da el que sigue (El número más uno).

Vemos que la definición es muy parecida a la de listas (La pondremos más explícita abajo), y que esta nos sirve para poder crear cualquier número natural. ¿Cómo funciona? Pues sabamos que el cero (0) cumple ser un número natural (Y es el más pequeño que podemos crear), por lo tanto si el cero es natural, entonces su sucesor es natural, por lo tanto el uno es natural.

Ahora bien, dado un número cualquiera, ¿Cómo sabemos que es natural? Pues si este número n es el sucesor de otro número m, entonces es natural, ¿Y cómo sabemos que m es natural? pues tendríamos que repetir este proceso hasta llegar a lo que conocemos como *caso base*, el cual es el mínimo número que podemos representar, por lo tanto m es un natural si y solo si, existen m sucesores a partir del cero.

Teniendo todo esto en cuenta, podemos dar una definición muy parecida para las listas, diremos que esta es *Una definición recursiva para listas* (Pues pueden existir otras):

1. La lista vacía es una lista.
2. Un elemento concatenado a una lista, es una lista.
3. Son todas.

Ya hemos usado definiciones recursivas, por ejemplo la práctica, las proposiciones eran recursivas, una proposición puede verse como un operador (`Conj`, `Neg`, `Disy`, `Impl`, `Syss`) y dos proposiciones más. Nuestro caso base es cuando tenemos una variable (Que es lo más pequeño en proposiciones que podemos crear).

2 Preguntas

Hemos creado nuestro propio tipo de dato Lista, su trabajo consiste en terminar las funciones señaladas. Es muy importante notar que no pueden usar las funciones `toHaskell` y `fromHaskell` para operar con las listas de haskell, esto quiere decir que por ejemplo para la función `reversa` está prohibido hacer lo siguiente:

```
reversa :: List a -> List a
reversa = fromHaskell $ reverse $ toHaskell
```

O algo parecido que permita operar con listas normales (Las de haskell). Es análogo para todas las demás funciones.

Aunque puedes trabajar con `deriving (Show)` en la entrega final debes crear la intancia para `Show`. Las listas tienen que verse de la siguiente manera:

```
Lista vacía: []
Lista [1]: (1:[])
Lista [1,2,3]: (1:(2:(3:[])))
```

3 Lineamientos

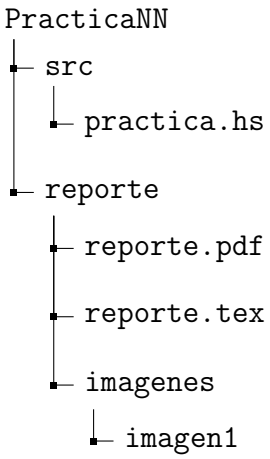
Puedes entregar tu práctica por medio de gitHub, tu repositorio tiene que ser un repositorio privado y tienes que añadirnos como colaboradores, los usuarios de gitHub son Rodrigo: `RodGpe` y Emiliano: `mildewyPrawn`. Tu repositorio deberá estar separado por carpetas y respetando la estructura de árbol que se muestra abajo. Además de que solo se revisará la rama *master* o *main*.

Si nadie del equipo está familiarizado con gitHub, pueden pedir ayuda a los ayudantes o entregarla de la forma usual, pero para las siguientes prácticas ya será la entrega obligatoria en gitHub.

3.1 Forma usual (Próxima a desaparecer)

Deberás entregar la práctica en el (correo/classroom), antes de las 23:59 del día especificado. El nombre del archivo deberá ser el nombre de su equipo y el sufijo “PNN” o “ESNN” según corresponda, donde NN debe ser reemplazado por el número de la práctica/ejercicio, además el archivo debe estar comprimido (.zip, .tar. gz, .tar.xz).

La carpeta descomprimida deberá verse de la siguiente manera:



No es necesaria la carpeta de imágenes, solo si utilizas alguna imágen, la cual puede estar en el formato que quieras.

Es importante que en el reporte también se incluyan los nombres de los/las integrantes del equipo.